Our Ref. No. 042390.P8112
Express Mail No.: EL466333061US

UNITED STATES PATENT APPLICATION

FOR

MANAGING ACCESSES IN A PROCESSOR

FOR ISOLATED EXECUTION

INVENTORS:

Carl M. Ellison
Roger A. Golliver
Howard C. Herbert
Derrick C. Lin
Francis X. McKeen
Gil Neiger
Ken Reneris
James A. Sutton
Shreekant S. Thakkar
Millind Mittal

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Blvd., 7th Floor
Los Angeles, CA 90025-1026
(714) 557-3800

BACKGROUND

1. Field of the Invention

This invention relates to microprocessors. In particular, the invention relates to processor security.

2. Description of Related Art

Advances in microprocessor and communication technologies have opened up many opportunities for applications that go beyond the traditional ways of doing business. Electronic commerce (E-commerce) and business-to-business (B2B) transactions are now becoming popular, reaching the global markets at a fast rate. Unfortunately, while modern microprocessor systems provide users convenient and efficient methods of doing business, communicating and transacting, they are also vulnerable to unscrupulous attacks. Examples of these attacks include virus, intrusion, security breach, and tampering, to name a few. Computer security, therefore, is becoming more and more important to protect the integrity of the computer systems and increase the trust of users.

Threats caused by unscrupulous attacks may be in a number of forms. Attacks may be remote without requiring physical accesses. An invasive remote-launched attack by hackers may disrupt the normal operation of a system connected to thousands or even millions of users. A virus program may corrupt code and/or data of a single-user platform.

Existing techniques to protect against attacks have a number of drawbacks. Anti-virus programs can only scan and detect known viruses. Most anti-virus programs use a weak policy in which a file or program is assumed good until proved bad. For many

security applications, this weak policy may not be appropriate. In addition, most anti-virus programs are used locally where they are resident in the platform. This may not be suitable in a group work environment. Security co-processors or smart cards using cryptographic or other security techniques have limitations in speed performance, memory capacity, and flexibility. Redesigning operating systems creates software compatibility issues and causes tremendous investment in development efforts.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1A is a diagram illustrating a logical operating architecture according to one embodiment of the invention.

Figure 1B is a diagram illustrating accessibility of various elements in the operating system and the processor according to one embodiment of the invention.

Figure 1C is a diagram illustrating a computer system in which one embodiment of the invention can be practiced.

Figure 2A is a diagram illustrating the isolated execution circuit shown in Figure 1C according to one embodiment of the invention.

Figure 2B is a diagram illustrating the access manager shown in Figure 2A according to one embodiment of the invention.

Figure 3A is a diagram illustrating the access checking circuit to manage snoop accesses according to one embodiment of the invention.

Figure 3B is a diagram illustrating the access checking circuit to manage process logical processor operations according to another embodiment of the invention.

Figure 4 is a flowchart illustrating a process to manage snoop accesses for isolated execution according to one embodiment of the invention.

Figure 5 is a flowchart illustrating a process to manage process logical processor operations for isolated execution according to one embodiment of the invention.

DESCRIPTION

In one embodiment of the present invention, a technique is provided to manage accesses in a processor for isolated execution. A configuration storage contains configuration parameters to configure an access transaction generated by a processor. The processor has a normal execution mode and an isolated execution mode. The access transaction has access information. An access checking circuit checks the access transaction using at least one of the configuration parameters and the access information.

The configuration parameters include an isolated setting and an execution mode word. The access information includes a physical address and an access type which indicates if the access transaction is a memory access, an input/output access, or a logical processor access. The logical processor access is one of a logical processor entry and a logical processor exit. The configuration storage includes a setting storage to contain the isolated setting for defining an isolated memory area corresponding to a memory external to the processor. The setting storage includes a base register, a mask register, and a length register to store a base value, a mask value and a length value, respectively. The base, mask, and length values form the isolated setting. The configuration storage further includes a processor control register to contain the execution mode word which is asserted when the processor is configured in the isolated execution mode.

In one embodiment, the access checking circuit includes an address detector to detect if the physical address is within the isolated memory area defined by the isolated setting. Two access checks are performed: one is on the physical address provided by the translation lookaside buffer (TLB) and another is on the physical address snooped on the front side bus (FSB). The address detector generates a processor isolated access signal. The access checking circuit further includes a snoop checking circuit to generate a processor snoop access signal. The snoop checking circuit includes a snoop combiner to

combine a cache access signal, the FSB isolated access signal, and an external isolated access signal from another processor. The combined cache access signal, the FSB isolated access signal and the external isolated access signal correspond to the processor snoop access signal. The access checking circuit further includes an access grant generator to generate an access grant signal indicating if the access transaction is valid.

In another embodiment, the access checking circuit includes a logical processor manager to manage a logical processor operation caused by the logical processor access. The logical processor manager includes (1) a logical processor register to store a logical processor count indicating a number of logical processor currently enabled, (2) a state enabler to enable a logical processor state when the logical processor access is valid, (3) a updater to update the logical processor count according to the logical processor access, the logical processor updater is enabled by the enabled logical processor state, (4) a minimum detector to determine if the logical processor count is equal to a minimum logical processor value, and (5) a maximum detector to determine if the logical processor count exceeds a maximum logical processor value. The logical processor updater initializes the logical processor register when there is no enabled logical processor. The logical processor updater updates the logical processor count in a first direction when the access transaction corresponds to the logical processor entry, and updates the logical processor count in a second direction opposite to the first direction when the access transaction corresponds to the logical processor exit. When the logical processor count is equal to the minimum logical processor value, the logical processor manager causes the processor to initialize the cache memory and the isolated setting from all isolated information. When the logical processor count exceeds the maximum logical processor value, the logical processor manager causes the processor to generate a failure or fault condition.

In the following description, for purposes of explanation, numerous details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. In other instances, well-known electrical structures and circuits are shown in block diagram form in order not to obscure the present invention.

ARCHITECTURE OVERVIEW

One principle for providing security in a computer system or platform is the concept of an isolated execution architecture. The isolated execution architecture includes logical and physical definitions of hardware and software components that interact directly or indirectly with an operating system of the computer system or platform. An operating system and the processor may have several levels of hierarchy, referred to as rings, corresponding to various operational modes. A ring is a logical division of hardware and software components that are designed to perform dedicated tasks within the operating system. The division is typically based on the degree or level of security and/or protection. For example, a ring-0 is the innermost ring, being at the highest level of the hierarchy. Ring-0 encompasses the most critical, security-sensitive components. In addition, modules in Ring-0 can also access to lesser privileged data, but not vice versa. Ring-3 is the outermost ring, being at the lowest level of the hierarchy. Ring-3 typically encompasses users or applications level and has the least security protection. Ring-1 and ring-2 represent the intermediate rings with decreasing levels of security and/or protection.

Figure 1A is a diagram illustrating a logical operating architecture 50 according to one embodiment of the invention. The logical operating architecture 50 is an abstraction of the components of an operating system and the processor. The logical operating

architecture 50 includes ring-0 10, ring-1 20, ring-2 30, ring-3 40, and a processor nub loader 52. The processor nub loader 52 is an instance of a processor executive (PE) handler. The PE handler is used to handle and/or manage a processor executive (PE) as will be discussed later. The logical operating architecture 50 has two modes of operation: normal execution mode and isolated execution mode. Each ring in the logical operating architecture 50 can operate in both modes. The processor nub loader 52 operates only in the isolated execution mode.

Ring-0 10 includes two portions: a normal execution Ring-0 11 and an isolated execution Ring-0 15. The normal execution Ring-0 11 includes software modules that are critical for the operating system, usually referred to as kernel. These software modules include primary operating system (e.g., kernel) 12, software drivers 13, and hardware drivers 14. The isolated execution Ring-0 15 includes an operating system (OS) nub 16 and a processor nub 18. The OS nub 16 and the processor nub 18 are instances of an OS executive (OSE) and processor executive (PE), respectively. The OSE and the PE are part of executive entities that operate in a secure environment associated with the isolated area 70 and the isolated execution mode. The processor nub loader 52 is a protected bootstrap loader code held within a chipset in the system and is responsible for loading the processor nub 18 from the processor or chipset into an isolated area as will be explained later.

Similarly, ring-1 20, ring-2 30, and ring-3 40 include normal execution ring-1 21, ring-2 31, ring-3 41, and isolated execution ring-1 25, ring-2 35, and ring-3 45, respectively. In particular, normal execution ring-3 includes N applications 42_1 to 42_N and isolated execution ring-3 includes K applets 46_1 to 46_K.

One concept of the isolated execution architecture is the creation of an isolated region in the system memory, referred to as an isolated area, which is protected by both

the processor and chipset in the computer system. The isolated region may also be in cache memory, protected by a translation lookaside buffer (TLB) access check. Access to this isolated region is permitted only from a front side bus (FSB) of the processor, using special bus (e.g., memory read and write) cycles, referred to as isolated read and write cycles. The special bus cycles are also used for snooping. The isolated read and write cycles are issued by the processor executing in an isolated execution mode. The isolated execution mode is initialized using a privileged instruction in the processor, combined with the processor nub loader 52. The processor nub loader 52 verifies and loads a ring-0 nub software module (e.g., processor nub 18) into the isolated area. The processor nub 18 provides hardware-related services for the isolated execution.

One task of the processor nub 18 is to verify and load the ring-0 OS nub 16 into the isolated area, and to generate the root of a key hierarchy unique to a combination of the platform, the processor nub 18, and the operating system nub 16. The operating system nub 16 provides links to services in the primary OS 12 (e.g., the unprotected segments of the operating system), provides page management within the isolated area, and has the responsibility for loading ring-3 application modules 45, including applets 461 to 46K, into protected pages allocated in the isolated area. The operating system nub 16 may also load ring-0 supporting modules.

The operating system nub 16 may choose to support paging of data between the isolated area and ordinary (e.g., non-isolated) memory. If so, then the operating system nub 16 is also responsible for encrypting and hashing the isolated area pages before evicting the page to the ordinary memory, and for checking the page contents upon restoration of the page. The isolated mode applets 46_1 to 46_K and their data are tamper- and monitor-proof from all software attacks from other applets, as well as from non-isolated-space applications (e.g., 42_1 to 42_N), dynamic link libraries (DLLs), drivers and

even the primary operating system 12. Only the processor nub 18 or the operating system nub 16 can interfere with or monitor the applet's execution.

Figure 1B is a diagram illustrating accessibility of various elements in the operating system 10 and the processor according to one embodiment of the invention. For illustration purposes, only elements of ring-0 10 and ring-3 40 are shown. The various elements in the logical operating architecture 50 access an accessible physical memory 60 according to their ring hierarchy and the execution mode.

The accessible physical memory 60 includes an isolated area 70 and a non-isolated area 80. The isolated area 70 includes applet pages 72 and nub pages 74. The non-isolated area 80 includes application pages 82 and operating system pages 84. The isolated area 70 is accessible only to elements of the operating system and processor operating in isolated execution mode. The non-isolated area 80 is accessible to all elements of the ring-0 operating system and to the processor.

The normal execution ring-0 11 including the primary OS 12, the software drivers 13, and the hardware drivers 14, can access both the OS pages 84 and the application pages 82. The normal execution ring-3, including applications 421 to 42N, can access only to the application pages 82. Both the normal execution ring-0 11 and ring-3 41, however, cannot access the isolated area 70.

The isolated execution ring-0 15, including the OS nub 16 and the processor nub 18, can access to both of the isolated area 70, including the applet pages 72 and the nub pages 74, and the non-isolated area 80, including the application pages 82 and the OS pages 84. The isolated execution ring-3 45, including applets 46_1 to 46_K, can access only to the application pages 82 and the applet pages 72. The applets 46_1 to 46_K reside in the isolated area 70.

Figure 1C is a diagram illustrating a computer system 100 in which one embodiment of the invention can be practiced. The computer system 100 includes a processor 110, a host bus 120, a memory controller hub (MCH) 130, a system memory 140, an input/output controller hub (ICH) 150, a non-volatile memory, or system flash, 160, a mass storage device 170, input/output devices 175, a token bus 180, a motherboard (MB) token 182, a reader 184, and a token 186. The MCH 130 may be integrated into a chipset that integrates multiple functionalities such as the isolated execution mode, host-to-peripheral bus interface, memory control. Similarly, the ICH 150 may also be integrated into a chipset together or separate from the MCH 130 to perform I/O functions. For clarity, not all the peripheral buses are shown. It is contemplated that the system 100 may also include peripheral buses such as Peripheral Component Interconnect (PCI), accelerated graphics port (AGP), Industry Standard Architecture (ISA) bus, and Universal Serial Bus (USB), etc.

The processor 110 represents a central processing unit of any type of architecture, such as complex instruction set computers (CISC), reduced instruction set computers (RISC), very long instruction word (VLIW), or hybrid architecture. In one embodiment, the processor 110 is compatible with an Intel Architecture (IA) processor, such as the Pentium™ series, the IA-32™ and the IA-64™. The processor 110 includes a normal execution mode 112 and an isolated execution circuit 115. The normal execution mode 112 is the mode in which the processor 110 operates in a non-secure environment, or a normal environment without the security features provided by the isolated execution mode. The isolated execution circuit 115 provides a mechanism to allow the processor 110 to operate in an isolated execution mode. The isolated execution circuit 115 provides hardware and software support for the isolated execution mode. This support includes configuration for isolated execution, definition of an isolated area, definition (e.g.,

decoding and execution) of isolated instructions, generation of isolated access bus cycles, and generation of isolated mode interrupts.

In one embodiment, the computer system 100 can be a single processor system, such as a desktop computer, which has only one main central processing unit, e.g. processor 110. In other embodiments, the computer system 100 can include multiple processors, e.g. processors 110, 110a, 110b, etc., as shown in Figure 1C. Thus, the computer system 100 can be a multi-processor computer system having any number of processors. For example, the multi-processor computer system 100 can operate as part of a server or workstation environment. The basic description and operation of processor 110 will be discussed in detail below. It will be appreciated by those skilled in the art that the basic description and operation of processor 110 applies to the other processors 110a and 110b, shown in Figure 1C, as well as any number of other processors that may be utilized in the multi-processor computer system 100 according to one embodiment of the present invention.

The processor 110 may also have multiple logical processors. A logical processor, sometimes referred to as a thread, is a functional unit within a physical processor having an architectural state and physical resources allocated according to some partitioning policy. Within the context of the present invention, the terms "thread" and "logical processor" are used to mean the same thing. A multi-threaded processor is a processor having multiple threads or multiple logical processors. A multi-processor system (e.g., the system comprising the processors 110, 110a, and 110b) may have multiple multi-threaded processors.

The host bus 120 provides interface signals to allow the processor 110 or processors 110, 100a, and 110b to communicate with other processors or devices, e.g., the MCH 130. In addition to normal mode, the host bus 120 provides an isolated access

bus mode with corresponding interface signals for memory read and write cycles when the processor 110 is configured in the isolated execution mode. The isolated access bus mode is asserted on memory accesses initiated while the processor 110 is in the isolated execution mode. The isolated access bus mode is also asserted on instruction pre-fetch and cache write-back cycles if the address is within the isolated area address range and the processor 110 is initialized in the isolated execution mode. The processor 110 responds to snoop cycles to a cached address within the isolated area address range if the isolated access bus cycle is asserted and the processor 110 is initialized into the isolated execution mode.

The MCH 130 provides control and configuration of memory and input/output devices such as the system memory 140 and the ICH 150. The MCH 130 provides interface circuits to recognize and service isolated access assertions on memory reference bus cycles, including isolated memory read and write cycles. In addition, the MCH 130 has memory range registers (e.g., base and length registers) to represent the isolated area in the system memory 140. Once configured, the MCH 130 aborts any access to the isolated area that does not have the isolated access bus mode asserted.

The system memory 140 stores system code and data. The system memory 140 is typically implemented with dynamic random access memory (DRAM) or static random access memory (SRAM). The system memory 140 includes the accessible physical memory 60 (shown in Figure 1B). The accessible physical memory includes a loaded operating system 142, the isolated area 70 (shown in Figure 1B), and an isolated control and status space 148. The loaded operating system 142 is the portion of the operating system that is loaded into the system memory 140. The loaded OS 142 is typically loaded from a mass storage device via some boot code in a boot storage such as a boot read only memory (ROM). The isolated area 70, as shown in Figure 1B, is the memory area that is defined by the processor 110 when operating in the isolated execution mode.

Access to the isolated area 70 is restricted and is enforced by the processor 110 and/or the MCH 130 or other chipset that integrates the isolated area functionalities. The isolated control and status space 148 is an input/output (I/O)-like, independent address space defined by the processor 110 and/or the MCH 130. The isolated control and status space 148 contains mainly the isolated execution control and status registers. The isolated control and status space 148 does not overlap any existing address space and is accessed using the isolated bus cycles. The system memory 140 may also include other programs or data which are not shown.

The ICH 150 represents a known single point in the system having the isolated execution functionality. For clarity, only one ICH 150 is shown. The system 100 may have many ICHs similar to the ICH 150. When there are multiple ICHs, a designated ICH is selected to control the isolated area configuration and status. In one embodiment, this selection is performed by an external strapping pin. As is known by one skilled in the art, other methods of selecting can be used, including using programmable configuring registers. The ICH 150 has a number of functionalities that are designed to support the isolated execution mode in addition to the traditional I/O functions. In particular, the ICH 150 includes an isolated bus cycle interface 152, the processor nub loader 52 (shown in Figure 1A), a digest memory 154, a cryptographic key storage 155, an isolated execution logical processor manager 156, and a token bus interface 159.

The isolated bus cycle interface 152 includes circuitry to interface to the isolated bus cycle signals to recognize and service isolated bus cycles, such as the isolated read and write bus cycles. The processor nub loader 52, as shown in Figure 1A, includes a processor nub loader code and its digest (e.g., hash) value. The processor nub loader 52 is invoked by execution of an appropriate isolated instruction (e.g., IsoCreate) and is transferred to the isolated area 70. From the isolated area 80, the processor nub loader 52 copies the processor nub 18 from the system flash memory (e.g., the processor nub code

18 in non-volatile memory 160) into the isolated area 70, verifies and logs its integrity, and manages a symmetric key used to protect the processor nub's secrets. In one embodiment, the processor nub loader 52 is implemented in read only memory (ROM). For security purposes, the processor nub loader 52 is unchanging, tamper-proof and non-substitutable. The digest memory 154, typically implemented in RAM, stores the digest (e.g., hash) values of the loaded processor nub 18, the operating system nub 16, and any other critical modules (e.g., ring-0 modules) loaded into the isolated execution space. The cryptographic key storage 155 holds a symmetric encryption/decryption key that is unique for the platform of the system 100. In one embodiment, the cryptographic key storage 155 includes internal fuses that are programmed at manufacturing. Alternatively, the cryptographic key storage 155 may also be created with a random number generator and a strap of a pin. The isolated execution logical processor manager 156 manages the operation of logical processors operating in isolated execution mode. In one embodiment, the isolated execution logical processor manager 156 includes a logical processor count register that tracks the number of logical processors participating in the isolated execution mode. The token bus interface 159 interfaces to the token bus 180. A combination of the processor nub loader digest, the processor nub digest, the operating system nub digest, and optionally additional digests, represents the overall isolated execution digest, referred to as isolated digest. The isolated digest is a fingerprint identifying the ring-0 code controlling the isolated execution configuration and operation. The isolated digest is used to attest or prove the state of the current isolated execution.

The non-volatile memory 160 stores non-volatile information. Typically, the non-volatile memory 160 is implemented in flash memory. The non-volatile memory 160 includes the processor nub 18. The processor nub 18 provides the initial set-up and low-level management of the isolated area 70 (in the system memory 140), including verification, loading, and logging of the operating system nub 16, and the management of

the symmetric key used to protect the operating system nub's secrets. The processor nub 18 may also provide application programming interface (API) abstractions to low-level security services provided by other hardware. The processor nub 18 may also be distributed by the original equipment manufacturer (OEM) or operating system vendor (OSV) via a boot disk.

The mass storage device 170 stores archive information such as code (e.g., processor nub 18), programs, files, data, applications (e.g., applications 42_1 to 42_N), applets (e.g., applets 46_1 to 46_K) and operating systems. The mass storage device 170 may include compact disk (CD) ROM 172, floppy diskettes 174, and hard drive 176, and any other magnetic or optical storage devices. The mass storage device 170 provides a mechanism to read machine-readable media. When implemented in software, the elements of the present invention are the code segments to perform the necessary tasks. The program or code segments can be stored in a processor readable medium or transmitted by a computer data signal embodied in a carrier wave, or a signal modulated by a carrier, over a transmission medium. The "processor readable medium" may include any medium that can store or transfer information. Examples of the processor readable medium include an electronic circuit, a semiconductor memory device, a ROM, a flash memory, an erasable programmable ROM (EPROM), a floppy diskette, a compact disk CD-ROM, an optical disk, a hard disk, a fiber optical medium, a radio frequency (RF) link, etc. The computer data signal may include any signal that can propagate over a transmission medium such as electronic network channels, optical fibers, air, electromagnetic, RF links, etc. The code segments may be downloaded via computer networks such as the Internet, an Intranet, etc.

I/O devices 175 may include any I/O devices to perform I/O functions. Examples of I/O devices 175 include a controller for input devices (e.g., keyboard, mouse, trackball,

pointing device), media card (e.g., audio, video, graphics), a network card, and any other peripheral controllers.

The token bus 180 provides an interface between the ICH 150 and various tokens in the system. A token is a device that performs dedicated input/output functions with security functionalities. A token has characteristics similar to a smart card, including at least one reserved-purpose public/private key pair and the ability to sign data with the private key. Examples of tokens connected to the token bus 180 include a motherboard token 182, a token reader 184, and other portable tokens 186 (e.g., smart card). The token bus interface 159 in the ICH 150 connects through the token bus 180 to the ICH 150 and ensures that when commanded to prove the state of the isolated execution, the corresponding token (e.g., the motherboard token 182, the token 186) signs only valid isolated digest information. For purposes of security, the token should be connected to the digest memory.

MANAGING ACCESSES IN ISOLATED EXECUTION MODE

Figure 2A is a diagram illustrating the isolated execution circuit 115 shown in Figure 1C according to one embodiment of the invention. The isolated execution circuit 115 includes a core execution circuit 205, an access manager 220, and a cache memory manager 230.

The core execution unit 205 includes an instruction decoder and execution unit 210 and a translation lookaside buffer (TLB) 218. The instruction decoder and execution unit 210 receives an instruction stream 215 from an instruction fetch unit. The instruction stream 215 includes a number of instructions. The instruction decoder and execution unit 210 decodes the instructions and executes the decoded instructions. These instructions may be at the micro- or macro- level. The instruction decoder and execution unit 210 may be a physical circuit or an abstraction of a process of decoding and execution of

instructions. In addition, the instructions may include isolated instructions and non-isolated instructions. The instruction decoder and execution unit 210 generates a virtual address 212 when there is an access transaction caused by executing the instructions. The TLB 218 translates the virtual address 212 into a physical address.

The core execution circuit 205 interfaces with the access manager 220 via control/status information 222, operand 224, and access information 226. The control/status information 222 includes control bits to manipulate various elements in the access manager 220 and status data from the access manager 220. The operand 224 includes data to be written to and read from the access manager 220. The access information 226 includes address (e.g., the physical address provided by the TLB 218), read/write, and access type information.

The access manager 220 receives and provides the control/status information 222, the operand 224, receives the access information 226 from the core execution circuit 205 as a result of instruction execution, receives a cache access signal 235 from the cache memory manager 230, and receives an external isolated access signal 278 from another processor in the system. The external isolated access signal 278 is asserted when another processor in the system attempts to access the isolated memory area. The access manager 220 generates a processor isolated access signal 272, an access grant signal 274, and a processor snoop access signal 276. The processor isolated access signal 272 may be used to generate an isolated bus cycle sent to devices (e.g., chipsets) external to the processor 110 to indicate that the processor 110 is executing an isolated mode instruction. The processor snoop access signal 276 may be used by other devices or chipsets to determine if a snoop access is a hit or a miss. The processor isolated access signal 272, the access grant signal 274, and the processor snoop access signal 276 may also be used internally by the processor 110 to control and monitor other isolated or non-isolated activities.

The cache memory manager 230 receives the access information 226 from the core execution circuit 205 and generates the cache access signal 235 to the access manager 220. The cache memory manager 230 includes a cache memory 232 to store cache information and other circuits to manage cache transactions as known by one skilled in the art. The cache access signal 235 indicates the result of the cache access. In one embodiment, the cache access signal 235 is a cache miss signal which is asserted when there is a cache miss from a cache access.

Figure 2B is a diagram illustrating the access manager shown in Figure 2A according to one embodiment of the invention. The access manager 220 includes a configuration storage 250 and an access checking circuit 270. The access manager 220 exchanges operand 224 with and receives the access information 226 from the core execution circuit 205 shown in Figure 2A. The access manager 220 also receives the cache access signal 235 from the cache manager 230 and the external isolated access signal 278 from another processor as shown in Figure 2A. The access information 226 includes a physical address 282, a read/write (RD/WR#) signal 284, and an access type 286. The access information 226 is generated during an access transaction by the processor 110. The access type 286 indicates a type of access, including a memory reference, an input/output (I/O) reference, and a logical processor access. The logical processor access includes a logical processor entry to an isolated enabled state, and a logical processor withdrawal from an isolated enabled state.

The configuration storage 250 contains configuration parameters to configure an access transaction generated by the processor 110. The processor 110 has a normal execution mode and an isolated execution mode. The access transaction has the access information 226 as discussed above. The configuration storage 250 receives the operand 224 from the instruction decoder and execution unit 210 (Figure 2A) and includes a processor control register 252 and an isolated setting register 260. The processor control

register 252 contains an execution mode word 253. The execution mode word 253 is asserted when the processor 110 is configured in the isolated execution mode. In one embodiment, the execution mode word 253 is a single bit indicating if the processor 110 is in the isolated execution mode. The isolated setting register 260 contains an isolated setting 268. The isolated setting 268 defines the isolated memory area (e.g., the isolated area 70 in the system memory 140 shown in Figure 1C). The isolated setting register 260 includes a mask register 262, a base register 264 and a length register 266. The mask register 262 contains an isolated mask value 263. The base register 264 contains an isolated base value 265. The length register 266 contains a length value 267. The isolated mask, base, and length values 263, 265, and 267 form the isolated setting 268 and are used to define the isolated memory area. The isolated memory area may be defined by using any combination of the mask, base, and length values 263, 265, and 267. For example, the base value 265 corresponds to the starting address of the isolated memory area, while the sum of the base value 265 and the length value 267 corresponds to the ending address of the isolated memory area.

The access checking circuit 270 checks the access transaction using at least one of the configuration parameters (e.g., the execution mode word 253, the isolated setting 268) and the access information 226. The access checking circuit 270 generates the processor isolated access signal 272, the access grant signal 274, and the processor snoop access signal 276 using at least one of the isolated area parameters in the configuration storage 250 and the access information 226 in a transaction generated by the processor 110 and an FSB physical address 228. The FSB physical address 228 is typically provided by another processor and is snooped on the FSB. The processor isolated access signal 272 is asserted when the processor 110 is configured in the isolated execution mode. The access grant signal 274 is used to indicate if an isolated access has been granted. The processor

snoop access signal 276 is used to determine if an isolated access results in a hit or a miss.

Figure 3A is a diagram illustrating the access checking circuit 270 to manage snoop accesses according to one embodiment of the invention. The access checking circuit 270 includes a TLB address detector 310 and an FSB address detector 336, an access grant generator 320, and a snoop checking circuit 330.

The TLB address detector 310 receives the isolated setting 268 (e.g., the isolated mask value 263, the isolated base value 265) from the configuration storage 250 in Figure 2B. The TLB address detector 310 detects if the physical address 282 is within the isolated memory area defined by the isolated setting 260. The TLB address detector 310 includes a masking element 312, a comparator 314, and a combiner 318. The masking element 312 masks the physical address 282 with the isolated mask value 263. In one embodiment, the masking element 312 performs a logical AND operation. The comparator 314 compares the result of the masking operation done by the masking element 312 and the isolated base value 265, and generates a comparison result. The combiner 318 combines the comparison result with other conditions to generate the processor isolated access signal 272. The processor isolated access signal 272 is asserted when the physical address 282 is within the isolated memory area as defined by the isolated mask and base values 263 and 265, respectively, and when other conditions are satisfied.

The FSB address detector 336 essentially performs similar tasks as the TLB address detector 310. It also has a masking element, a comparator, and a combiner. The address detector 336 receives the isolated setting 268 to detect if the FSB physical address 228 is within the isolated memory area defined by the isolated setting 260. The address detector 336 generates an FSB isolated access signal 339. In addition, the TLB

and FSB address detectors 310 and 336 may include other circuits to check the limit or size of the isolated memory area using the isolated length value 267.

The access grant generator 320 combines the processor isolated access signal 272 and the execution mode word 253 to generate an access grant signal 274. The access grant signal 274 is asserted when both the isolated access signal 272 and the execution mode word 253 are asserted to indicate that an isolated access is valid or allowed as configured. In one embodiment, the access grant generator 320 performs a logical AND operation. If an access to the isolated memory area is attempted and the access grant signal 274 is de-asserted, an failure or fault condition is generated.

The snoop checking circuit 330 generates the processor snoop access signal 276. The snoop checking circuit 330 includes a snoop combiner 340 to combine the cache access signal 235, the FSB isolated access signal 339, and an external isolated access signal 278 from another processor. The combined cache access signal 235, the FSB isolated access signal 339 and the external isolated access signal 278 correspond to the processor snoop access signal 276. In one embodiment, the snoop combiner 340 includes an exclusive-OR (X-OR) element 342 and an OR element 344. The X-OR element 342 performs an exclusive-OR operation on the isolated access signal 272 and the external isolated access 278. The OR element 344 performs an OR operation on the result of the X-OR element 342 and the cache access signal 235.

The snoop combiner 340 ensures proper functionality in a multiprocessor system when not all the processors have been initialized for isolated memory area accesses. For example, the configuration storage, including the isolated setting, may not yet be initialized. The X-OR element 342 ensures that a snoop hit can only occur from a processor that has been allowed for isolated access. If one processor is not yet participating in the isolated memory area accesses, it will not be able to snoop a line out

of another processor that is participating in the isolated memory area accesses. Similarly, a processor that has been enabled for isolated accesses will not inadvertently snoop a line out of another processor that has not yet been enabled.

The processor snoop access signal 276 is asserted indicating there is an access miss when the cache access signal 235 is asserted indicating there is a cache miss, or when the FSB isolated access signal 339 and the external isolated access signal 278 do not match.

Figure 3B is a diagram illustrating the access checking circuit 270 to manage process logical processor operations according to another embodiment of the invention. The access checking circuit 270 includes a logical processor manager 360.

A physical processor may have a number of logical processors, each has its own logical processor. Each logical processor may enter or exit a logical processor state, referred to as a logical processor access. A logical processor access is typically generated when the corresponding logical processor executes an isolated instruction, such as isolated enter (iso_enter) and isolated_exit (iso_exit). The logical processor manager 360 manages a logical processor operation caused by the logical processor access. Essentially, the logical processor manager 360 keeps track of the number of enabled logical processors in the processor. The logical processor manager 360 includes a logical processor register 370, a logical processor state enabler 382, a logical processor updater 380, a minimum detector 374, and a maximum detector 376. The logical processor register 370 store a logical processor count 372 to indicate a number of logical processors currently enabled. The logical processor state enabler 382 enables a logical processor state when the logical processor access is valid. The logical processor updater 380 updates the logical processor count 372 according to the logical processor access. The logical processor updater 380 is enabled by the enabled logical processor state. In

one embodiment, the logical processor register 370 and the logical processor updater 380 are implemented as an up/down counter with enable. The minimum detector 374 determines if the logical processor count 372 is equal to a minimum logical processor value (e.g., zero). The maximum detector 376 determines if the logical processor count 372 exceeds a maximum logical processor value. The maximum logical processor value is a number indicating the maximum number of logical processors that can be supported by the isolated execution mode in the processor 110.

The logical processor updater 380 initializes the logical processor register 370 when there is no enabled logical processor. The logical processor updater 380 updates the logical processor count 372 in a first direction (e.g., incrementing) when the access transaction corresponds to the logical processor entry. The logical processor updater 380 updates the logical processor count 372 in a second direction opposite to the first direction (e.g., decrementing) when the access transaction corresponds to the logical processor exit or a logical processor withdrawal. When the logical processor count 372 is equal to the minimum logical processor value, the logical processor manager 360 causes the processor 110 to initialize the cache memory 232 (Figure 2A) and the isolated setting register (Figure 2A) from all isolated information to restore the initial conditions in these storage elements. When the logical processor count 372 exceeds the maximum logical processor value, the logical processor manager 360 causes the processor 110 to generate a failure or fault condition because the total number of logical processors exceed the maximum number of logical processors that can be supported in the processor.

Figure 4 is a flowchart illustrating a process 400 to manage snoop accesses for isolated execution according to one embodiment of the invention.

Upon START, the process 400 defines an isolated memory area using the isolated setting (e.g., isolated mask and base values) (Block 410). Then, the process 400 asserts

the execution mode word in the processor control register to configure the processor in the isolated execution mode (Block 420). Then, the process 400 receives the access information from an access transaction by the processor (Block 425). The access information includes a physical address (as provided by the TLB) and an access type. Next, the process 400 determines if the physical address as generated in a transaction is within the isolated memory area as defined by the isolated setting (Block 430). If not, the process 400 generates a failure or fault condition (Block 435) and is then terminated. Otherwise, the process 400 asserts the isolated access signal (Block 440).

Next, the process 400 generates a processor snoop access signal by combining the cache access signal, the external isolated access signal, and the processor isolated access signal. Then the process 400 is terminated.

Figure 5 is a flowchart illustrating a process 500 to manage process logical processor operations for isolated execution according to one embodiment of the invention.

Upon START, the process 500 initializes the logical processor register when there is no enabled logical processor (Block 510). Then the process 500 executes a logical processor access instruction (e.g., iso_enter, iso_exit). The logical processor access instruction asserts the execution mode word. Next, the process 500 enables the logical processor state (Block 525). Then, the process 500 determines the logical processor access type (Block 530).

If the logical processor access type is a logical processor entry, the process 500 updates the logical processor count in a first direction (e.g., incrementing) (Block 540). Then, the process 500 determines if the logical processor count exceeds the maximum logical processor value (Block 550). If not, the process 500 goes to block 570.

Otherwise, the process 500 generates a failure or fault condition (Block 560) and is then terminated.

If the logical processor access type is a logical processor exit or logical processor withdrawal, the process 500 updates the logical processor count in a second direction opposite to the first direction (e.g., decrementing) (Block 545). Then, the process 500 determines if the logical processor count is equal to the minimum value (e.g., zero) (Block 555). If not, the process 500 goes to block 570. Otherwise, the process 500 initializes the cache memory and the isolated setting register from all the isolated information (Block 565).

Next, the process 500 determines if there is a next logical processor access (Block 570). If there is a next logical processor access, the process 500 returns to block 520 to execute a logical processor access instruction. If there is no more logical processor access, the process 500 is terminated.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.